



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Московский государственный технический университет имени  
Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

---

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

---

## Отчет по лабораторной работе №7 по курсу «Экономика программной инженерии»

Тема Оценка параметров программного проекта с использованием СОСОМО II

---

Студент Кононенко С.С.

---

Группа ИУ7-83Б

---

Оценка (баллы)

---

Преподаватель Барышникова М.Ю.

---

# SOCOMO II

Модель SOCOMO I полностью основана на модели водопада, но из-за освоения объектно-ориентированного подхода в процессе разработки программного обеспечения SOCOMO I не дает точных результатов. Итак, чтобы преодолеть ограничения SOCOMO I, был разработан SOCOMO II.

Первоочередной целью модели SOCOMO II является создание возможностей поддержки для постоянного внесения поправок в модель и предоставление количественной аналитической структуры, методов и инструментов. Он также способен исследовать влияние усовершенствований программных технологий на жизненный цикл разработки программного обеспечения.

Модели оценки, включенные в SOCOMO II, включают модель композиции приложения, модель раннего проектирования, модель повторного использования и модель постархитектуры.

- **Модель композиции приложения.** Эта модель предназначена для использования с повторно используемыми компонентами и генерирует оценки разработки прототипа и работает на основе точек объекта. Модель больше подходит для разработки прототипа системы. Чтобы оценить общие усилия, выполняются следующие шаги.
- **Ранняя модель дизайна.** Модель, используемая на этапе проектирования системы после получения требований. Его оценки создаются на основе функциональных точек, которые затем переводятся в несколько строк исходного кода. Оценки на этом этапе основаны на основной формуле для моделей алгоритмов:

$$\text{Усилие} = A \times \text{Размер} \times B \times M$$

- **Модель повторного использования.** Это модель, которая вычисляет усилия, необходимые для объединения повторно используемых компонентов или программного кода, который спонтанно создается инструментами проектирования или преобразования программ. Есть

два типа повторно используемых кодов: черный ящик и код белого ящика. Код черного ящика используется, когда в нем нет понимания кода и модификации. И наоборот, белое поле используется при интеграции нового кода. Усилия, необходимые для интеграции этого кода, оцениваются следующим образом:

$$E = \frac{ALOC \times \frac{AT}{100}}{ATPROD}$$

- **Постархитектурная модель.** После разработки архитектуры системы можно сделать более точную оценку программного обеспечения. Эта модель считается самой подробной из всех моделей, которая может дать наиболее подробную и точную оценку. Усилия постархитектурной модели можно вычислить следующим образом:

$$E = A \times \text{Размер} \ B \times M$$

# Задание

Компания получила заказ на разработку клиентского мобильного приложения брокерской системы. Программа позволяет просматривать актуальную биржевую информацию, производить сделки и отслеживать их выполнение.

## Расчет по методу функциональных точек

**FTR** – количество связанных с каждым функциональным типом файлов типа ссылок.

**DET** – количество связанных с каждым функциональным типом элементарных данных.

**RET** – количество типов элементов записей.

**EI** (внешний ввод) – элементарный процесс, перемещающий данные из внешней среды в приложение.

**EO** (внешний вывод) – элементарный процесс, перемещающий данные, вычисленные в приложении, во внешнюю среду.

**EQ** (внешний запрос) – элементарный процесс, состоящий из комбинации «запрос/ответ», не связанный с вычислением производных данных или обновлением внутренних логических файлов (базы данных).

**ILF** (внутренний логический файл) – выделяемые пользователем логически связанные группы данных или блоки управляющей информации, которые поддерживаются внутри продукта и обслуживаются через внешние вводы.

**EIF** (внешний интерфейсный файл) – выделяемые пользователем логически связанные группы данных или блоки управляющей информации, на которые ссылается продукт, но которые поддерживаются вне продукта.

В нашем приложении используются 4 внутренних файла: таблица с логинами и паролями, таблица с типом заявки, именем бумаги, ценой и количеством, таблица с названием бумаги. Также существует одна внешняя таблица с информацией о бирже с названием бумаги, ценой и изменением.

Вычисление EI:

- Добавить бумагу

FTR = 1 (один внутренний логический файл)

DET = 2 (кнопка, название бумаги)

- Удалить заявку

FTR = 1

DET = 5 (тип, имя, цена, количество, кнопка)

- Изменить заявку

FTR = 1

DET = 5 (тип, имя, цена, количество, кнопка)

- Удалить заявку

FTR = 1

DET = 5 (тип, имя, цена, количество, кнопка)

Уровень сложности – низкий.

Вычисление E0:

- Вывод списка заявок

FTR = 1 (один внутренний логический файл)

DET = 4 (тип, имя, цена, количество)

- Вывод биржевых сводок

FTR = 2 (внутренний логический файл и внешний интерфейсный файл)

DET = 3 (имя, цена, изменения)

Уровень сложности – низкий.

Вычисление EQ:

- Запрос на авторизацию

FTR = 1

DET = 4 (логин, пароль, кнопка, флажок)

Уровень сложности – низкий.

Вычисление ILF:

- ILF

RET = 4 (элементы записи)

DET = 4 (элементы данных)

Уровень сложности – низкий.

Вычисление EIF:

- EIF

RET = 2 (элементы записи)

DET = 3 (элементы данных)

Уровень сложности – низкий.

После расчетов получен результат:

- Нормированное количество функциональных точек – 50.47
- Количество функциональных точек – 49
- Количество строк исходного кода – 3401

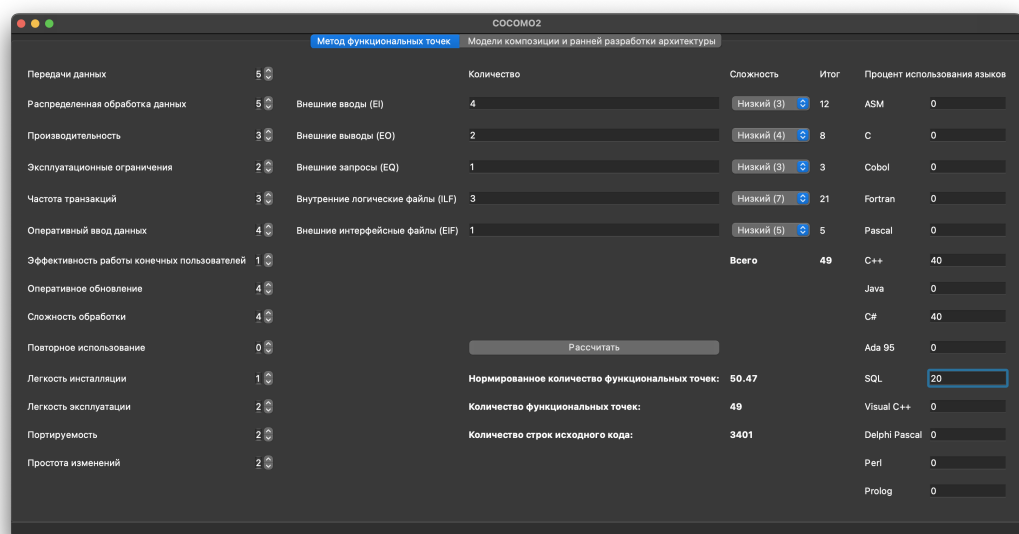


Рисунок 1 – Расчет проекта по методу функциональных точек

## Оценка по модели COSOMO II

Показатели проекта:

- Новизна проекта (PREC) – полное отсутствие прецедентов, полностью непредсказуемый проект (т.к. была сформирована новая команда разработчиков, только отдельные члены имели некоторый опыт создания систем подобного типа)
- Гибкость процесса разработки (FLEX) – по большей части согласованный процесс (график жесткий, точной регламентации нет)
- Разрешение рисков в архитектуре системы (RESL) – некоторое (40%)
- Сплоченность команды (TEAM) – некоторая согласованность (команда новая, но были проведены определенные мероприятия по сплочению)
- Уровень развития процесса разработки (PMAT) – уровень 1+ (только начинают внедрять)

$$p = 1.23$$

Факторы показателя степени модели	
Новизна проекта (PREC)	Почти полное отсутствие прецедентов, в значительной мере непредсказуемый проект
Гибкость процесса разработки (FLEX)	Большой частью согласованный процесс
Анализ архитектуры системы и рисков (RESL)	Некоторое (40 %)
Сплоченность команды (TEAM)	Некоторая согласованность
Уровень развития процесса разработки (PMAT)	Уровень 1+ СММ
Р равно:	1.2317
<input type="button" value="Рассчитать"/>	
Средняя зарплата	0

Рисунок 2 – Расчет показателя степени

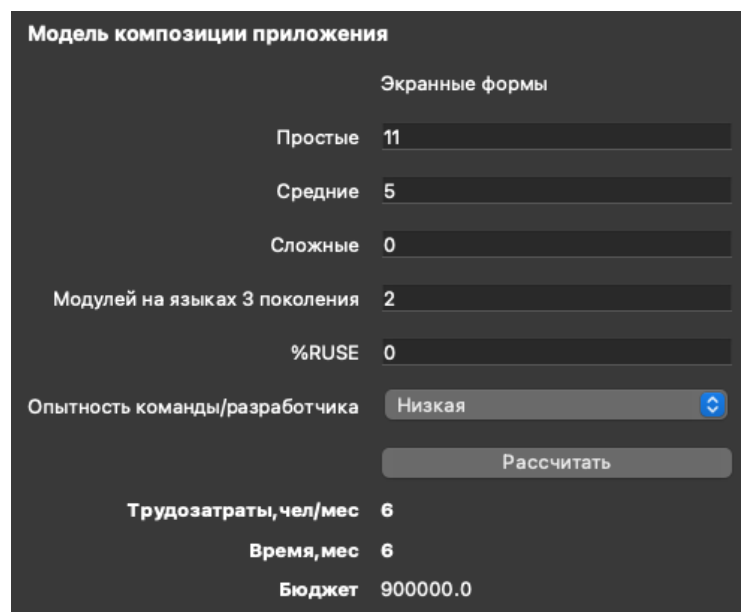
Композиция приложения:

- Страница авторизации – 3 простых поля и 1 средней сложности (обращение к БД)
- Страница биржевых сводок – 3 простых поля и 1 средней сложности (обращение к БД)
- Страница заявок – 1 простое поле и 2 средней сложности (обращение к БД)

- Страница новой заявки – 4 простых поля и 1 средней сложности (обращение к БД)

Итого:

- Простые поля = 11
- Поля средней сложности = 5
- Поля высокой сложности = 0
- Модули на ЯП третьего поколения = 2
- Повторное использование = 0%
- Опыт команды – низкий



Модель композиции приложения	
Экранные формы	
Простые	11
Средние	5
Сложные	0
Модулей на языках 3 поколения	2
%RUSE	0
Опытность команды/разработчика	Низкая
<b>Рассчитать</b>	
Трудозатраты, чел/мес	6
Время, мес	6
Бюджет	900000.0

Рисунок 3 – Модель композиции приложения (при средней зарплате 150 000)

Модель ранней разработки архитектуры:

- PERS (возможности персонала) – номинальный
- RCPX (надежность и уровень сложности разрабатываемой системы) – очень высокий
- RUSE (повторное использование компонентов) – низкий



- PDIF (сложность платформы разработки) – высокий
- PREX (опыт персонал) – низкий
- FCIL (средства поддержки) – очень высокий
- SCED (график работ) – очень высокий
- KSLOC = 3.5 (из метода функциональных точек)

**Модель ранней разработки архитектуры**

PERS Номинальный

RCPX Очень высокий

RUSE Низкий

PDIF Высокий

PREX Низкий

FCIL Очень высокий

SCED Очень высокий

Рассчитать

Трудозатраты, чел/мес 23

Время, мес 10

Бюджет 3450000.0

Рисунок 4 – Модель ранней разработки архитектуры (при средней зарплате 150 000)

## Выводы

В ходе выполнения работы был разработан инструмент для определения трудозатрат и времени разработки проекта методом COSOMO II. Был выполнен анализ выданного задания, а именно:

- рассчитаны функциональные точки
- рассчитан показатель степени модели (p)
- были определены факторы, влияющие на показатель степени

- произведен расчет трудозатрат и времени по модели ранней разработки архитектуры приложения и модели композиции приложения

В итоге было выяснено, что модель композиции приложения дает более оптимистичный прогноз, по сравнению с моделью ранней архитектуры приложения.